PORTFOLIO

Food Delivery Time Estimation

Github
Ignatius Evans Erlangga

INTRODUCTION

Porter is the largest intra-city logistics marketplace in India.

As a leader in the country's **$40 billion intra-city logistics market**, Porter aims to improve the livelihoods of over **150,000 driver partners** by providing them with consistent income opportunities and financial independence.

To date, the company has served more than **5 million customers**. Porter collaborates with a wide range of restaurants to deliver their goods directly to consumers.

PROJECT GOALS

This project aims to develop a delivery time estimation prediction model.

The dataset used focuses on food delivery operations in India.
Porter has 1.5 million driver partners and has served 5 million customers.

Therefore, **improving service quality**—particularly in relation to delivery time accuracy—is essential.

TABLE OF CONTENTS

Importing Libraries

01 Data Preparation & Preprocessing

Data Manipulation:

- import pandas as pd
- import numpy as np

Data Visualization:

- import matplotlib.pyplot as plt
- import seaborn as sns

Data Preprocessing:

- from sklearn.model_selection import train_test_split
- from sklearn.preprocessing import LabelEncoder, MinMaxScaler

Data Modelling:

- from sklearn.linear_model import LinearRegression
- from sklearn.svm import SVR
- from xgboost import XGBRegressor
- import xgboost as xgb
- from sklearn.neighbors import KNeighborsRegressor

Data Evaluation:

- from sklearn.model_selection import train_test_split
- from sklearn.metrics import mean_squared_error
- from sklearn.metrics import mean_absolute_error
- from sklearn.metrics import r2_score

Data Model Exporting:

- Import pickle

Loading the Data

01 Data Preparation & Preprocessing

```
data.head()
```

	market_id	created_at	actual_delivery_time	store_id	store_primary_category	order_protocol	total_items	subtotal	num_distinct_items	min_item_price	max_item_price	total_onshift_partners	total_busy_partners	total_outstanding_orders
0	1.0	2015-02-06 22:24:17	2015-02-06 23:27:16	df263d996281d984952c07998dc54358	american	1.0	4	3441	4.0	557.0	1239.0	33.0	14.0	21.0
1	2.0	2015-02-10 21:49:25	2015-02-10 22:56:29	f0ade77b43923b38237db569b016ba25	mexican	2.0	1	1900	1.0	1400.0	1400.0	1.0	2.0	2.0
2	3.0	2015-01-22 20:39:28	2015-01-22 21:09:09	f0ade77b43923b38237db569b016ba25	NaN	1.0	1	1900	1.0	1900.0	1900.0	1.0	0.0	0.0
3	3.0	2015-02-03 21:21:45	2015-02-03 22:13:00	f0ade77b43923b38237db569b016ba25	NaN	1.0	6	6900	5.0	600.0	1800.0	1.0	1.0	2.0
4	3.0	2015-02-15 02:40:36	2015-02-15 03:20:26	f0ade77b43923b38237db569b016ba25	NaN	1.0	3	3900	3.0	1100.0	1600.0	6.0	6.0	9.0

Descriptive Statistics

01 Data Preparation & Preprocessing

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 197428 entries, 0 to 197427
Data columns (total 14 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   market_id                196441 non-null  float64
 1   created_at               197428 non-null  object
 2   actual_delivery_time     197421 non-null  object
 3   store_id                 197428 non-null  object
 4   store_primary_category   192668 non-null  object
 5   order_protocol           196433 non-null  float64
 6   total_items              197428 non-null  int64
 7   subtotal                 197428 non-null  int64
 8   num_distinct_items       197428 non-null  int64
 9   min_item_price           197428 non-null  int64
 10  max_item_price           197428 non-null  int64
 11  total_onshift_partners   181166 non-null  float64
 12  total_busy_partners      181166 non-null  float64
 13  total_outstanding_orders 181166 non-null  float64
dtypes: float64(5), int64(5), object(4)
memory usage: 21.1+ MB
```

Data types for each column

Descriptive Statistics

01 Data Preparation & Preprocessing

```
data.describe()
```

	created_at	actual_delivery_time	total_items	subtotal	num_distinct_items	min_item_price	max_item_price
count	78474	78472	78474.000000	78474.000000	78473.000000	78473.000000	78473.000000
mean	2015-02-04 21:22:20.727208960	2015-02-04 22:12:03.785821696	3.246642	2726.084754	2.698979	689.728161	1180.300944
min	2014-10-19 05:24:15	2015-01-21 16:17:43	1.000000	0.000000	1.000000	-86.000000	0.000000
25%	2015-01-29 02:19:21.249999872	2015-01-29 03:09:36.249999872	2.000000	1448.000000	2.000000	299.000000	800.000000
50%	2015-02-05 03:17:07.500000	2015-02-05 04:25:30.500000	3.000000	2253.000000	2.000000	595.000000	1095.000000
75%	2015-02-12 01:30:09.249999872	2015-02-12 02:14:43.500000	4.000000	3455.000000	3.000000	945.000000	1400.000000
max	2015-02-18 06:00:44	2015-02-19 22:45:31	411.000000	24300.000000	20.000000	8999.000000	8999.000000
std	NaN	NaN	2.958795	1841.830626	1.646766	537.978962	583.636451

Descriptive Statistics for the Data

Descriptive Statistics

01 Data Preparation & Preprocessing

Porter delivery time estimation

Kolom (14):

- market_id = ID market (int) lokasi restoran
- created_at = Timestamp (tanggal dan waktu) pesanan dibuat
- actual_delivery_time = Timestamp (tanggal dan waktu) pesanan dikirimkan
- store_id = ID restoran
- store_primary_category = Kategori restoran
- order_protocol = Code value (int) untuk protokol pesanan (bagaimana pesanan dilakukan, misalnya dari Porter, menghubungi restoran langsung, pre-booked, pihak ketiga, dll.)
- total_items = Total jumlah barang yang dipesan
- subtotal = Harga akhir pesanan
- num_distinct_items = Jumlah barang yang berbeda (distinct) dalam pesanan
- min_item_price = Harga item terendah dalam pesanan
- max_item_price = Harga item tertinggi dalam pesanan
- total_onshift_partners = Jumlah mitra Porter yang sedang bertugas (stand by)
- total_busy_partners = Jumlah mitra Porter yang sedang menyelesaikan pesanan lain
- total_outstanding_orders = Jumlah pesanan yang harus diselesaikan at the moment

Beside is a description of each variable/feature in this dataset, which consists of 197,428 rows and 14 columns.

To predict delivery time duration,
the **dependent variable is defined as delivery_duration**,

while the remaining variables serve as independent variables acting as supporting features.

Exploratory Data Analysis

01 Data Preparation & Preprocessing

```
[6]  #Change dtype
     #to datetime
     data.created_at = pd.to_datetime(data.created_at)
     data.actual_delivery_time = pd.to_datetime(data.actual_delivery_time)

     #to object
     data.market_id = data.market_id.astype('object')
     data.store_primary_category = data.store_primary_category.astype ('object')
     data.order_protocol = data.order_protocol.astype('object')
```

```
[7]  #Duplicate and null values check

     print(f'Dataset dimensions\t: {data.shape}')
     print(f'Rows duplicated\t\t: {data.duplicated().sum()}')

     type_null = pd.DataFrame(data.dtypes).T.rename(index={0: 'Column Type'})
     type_null = pd.concat([type_null, pd.DataFrame(data.isnull().sum()).T.rename(index={0: 'Amount of Null Values'})])
     type_null = pd.concat([type_null, pd.DataFrame(round(data.isnull().sum()/data.shape[0]*100, 4)).T.rename(index={0: 'Percentage of Null Values'})])
     type_null = type_null.T
     type_null = type_null.reset_index().rename(columns={'index': 'feature'})
     type_null
```

Data type conversion

Duplicate and missing value checks

Exploratory Data Analysis

01 Data Preparation & Preprocessing

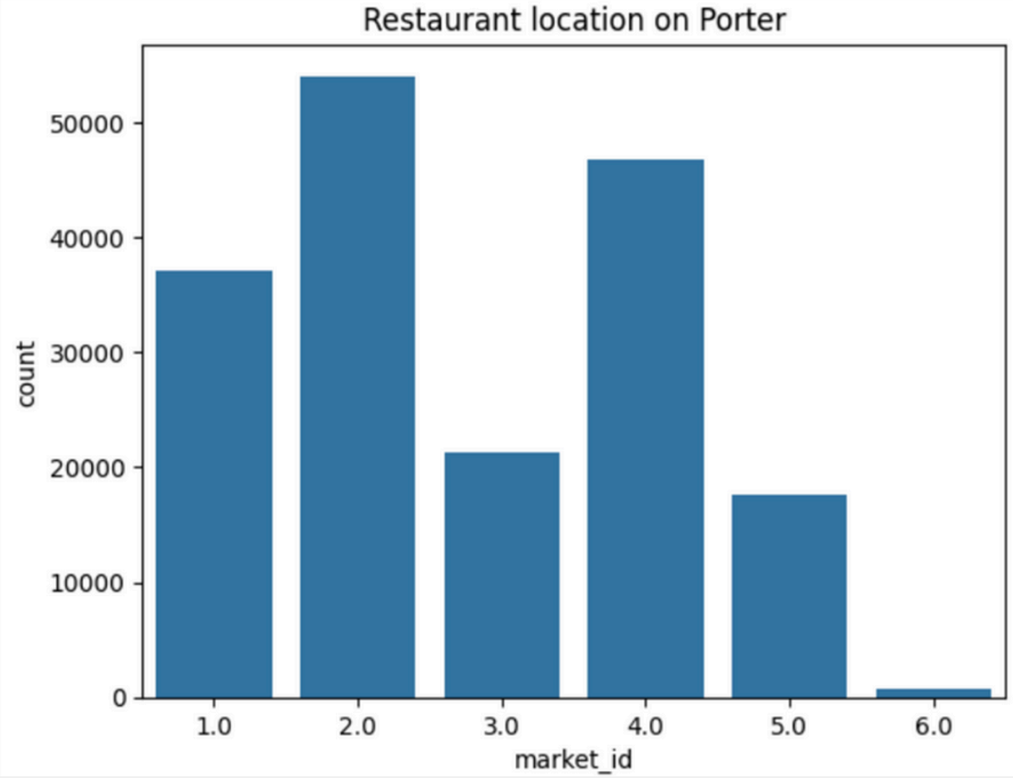
```
The data consist of 69 rows of data with total transaction equal to zero.
About 0.09% of total data

The data consist of 1019 rows of data with min item price quantity less/equal than zero.
About 1.30% of total data

The data consist of 6 rows of data with max item price equal to zero.
About 0.01% of total data

The data consist of 8 rows of data with current online Porter Partners that is less than zero.
About 0.01% of total data

The data consist of 12 rows of data with total busy Porter Partners that is less than zero.
About 0.02% of total data

The data consist of 12 rows of data with current ongoing orders less than zero.
About 0.02% of total data
```

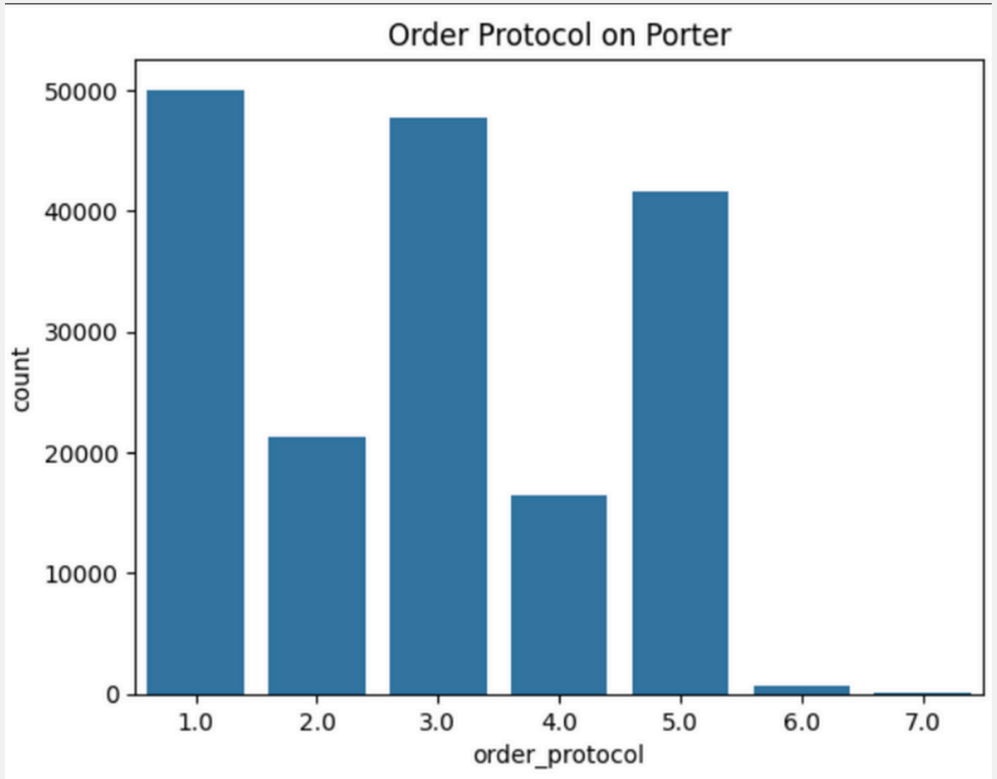
- Examining the percentage of data containing null values.
- Removing anomalous data.
- Dropping unnecessary columns.
- Adding required column names.

These steps are necessary to ensure the data is clean and properly structured.

Exploratory Data Analysis

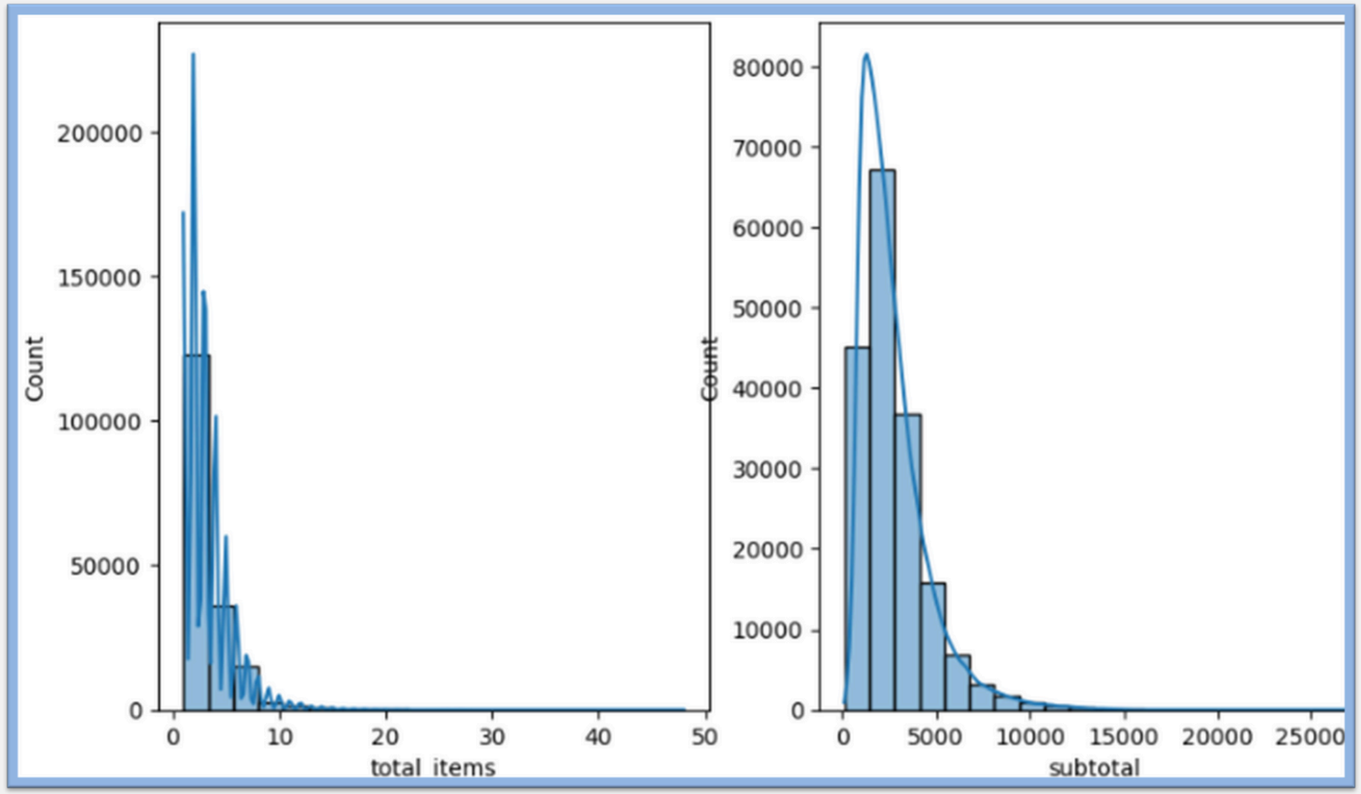
01 Data Preparation & Preprocessing





Exploratory Data Analysis

01 Data Preparation & Preprocessing

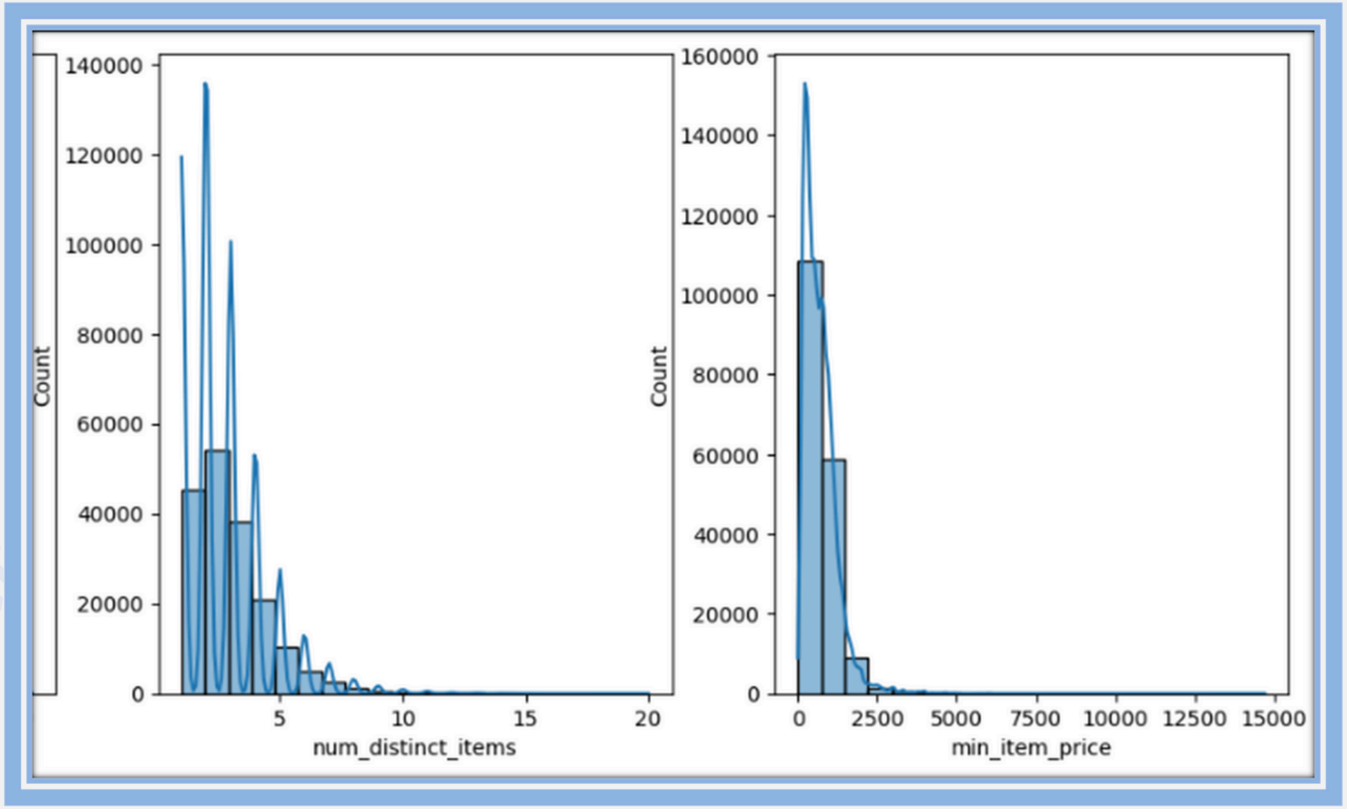


total_items:
On average, customers purchase 2 items per transaction.

subtotal:
Most transactions have a total cost of Rs. 2000.

Exploratory Data Analysis

01 Data Preparation & Preprocessing

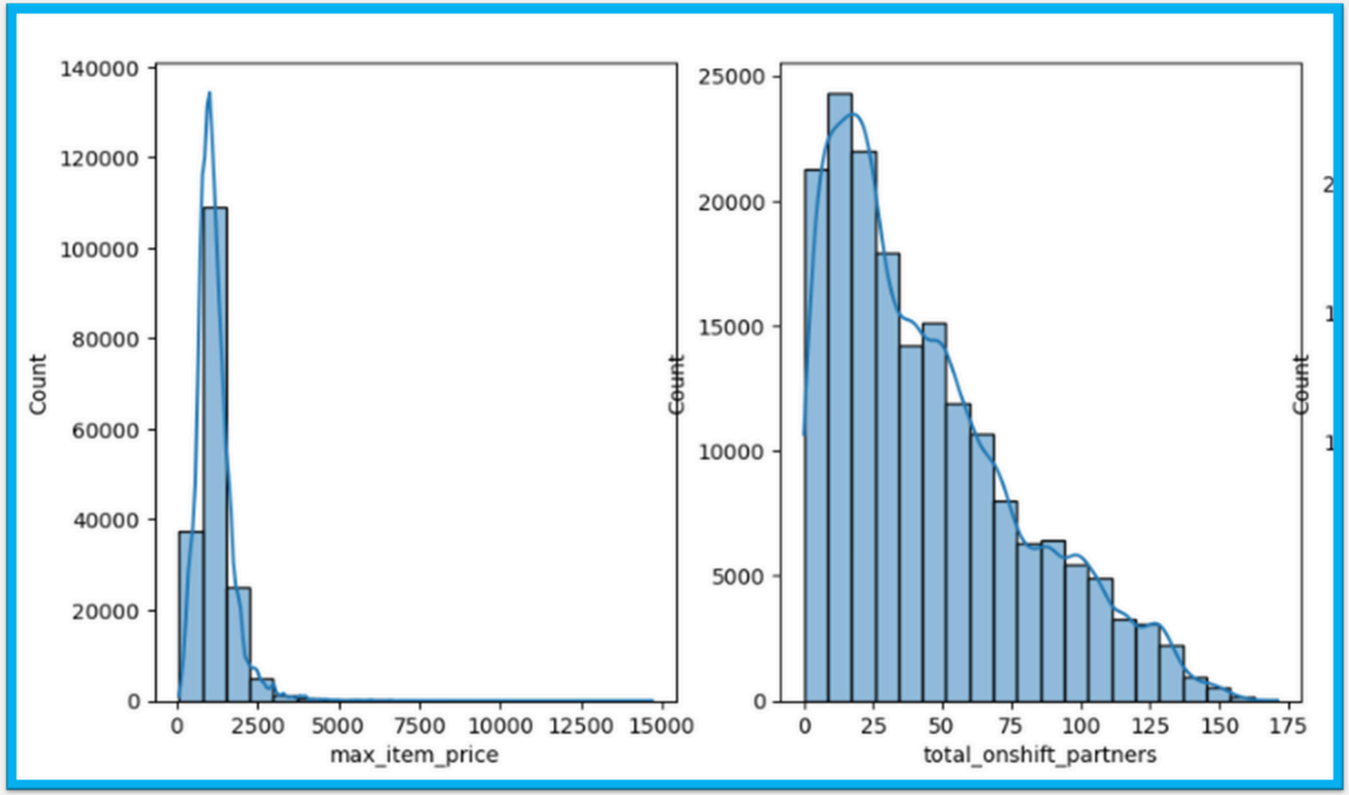


num_distinct_items:
Consumers typically order 2 distinct items per transaction.

min_items_price:
In general, the lowest item/food price is below Rs. 1000.

Exploratory Data Analysis

01 Data Preparation & Preprocessing

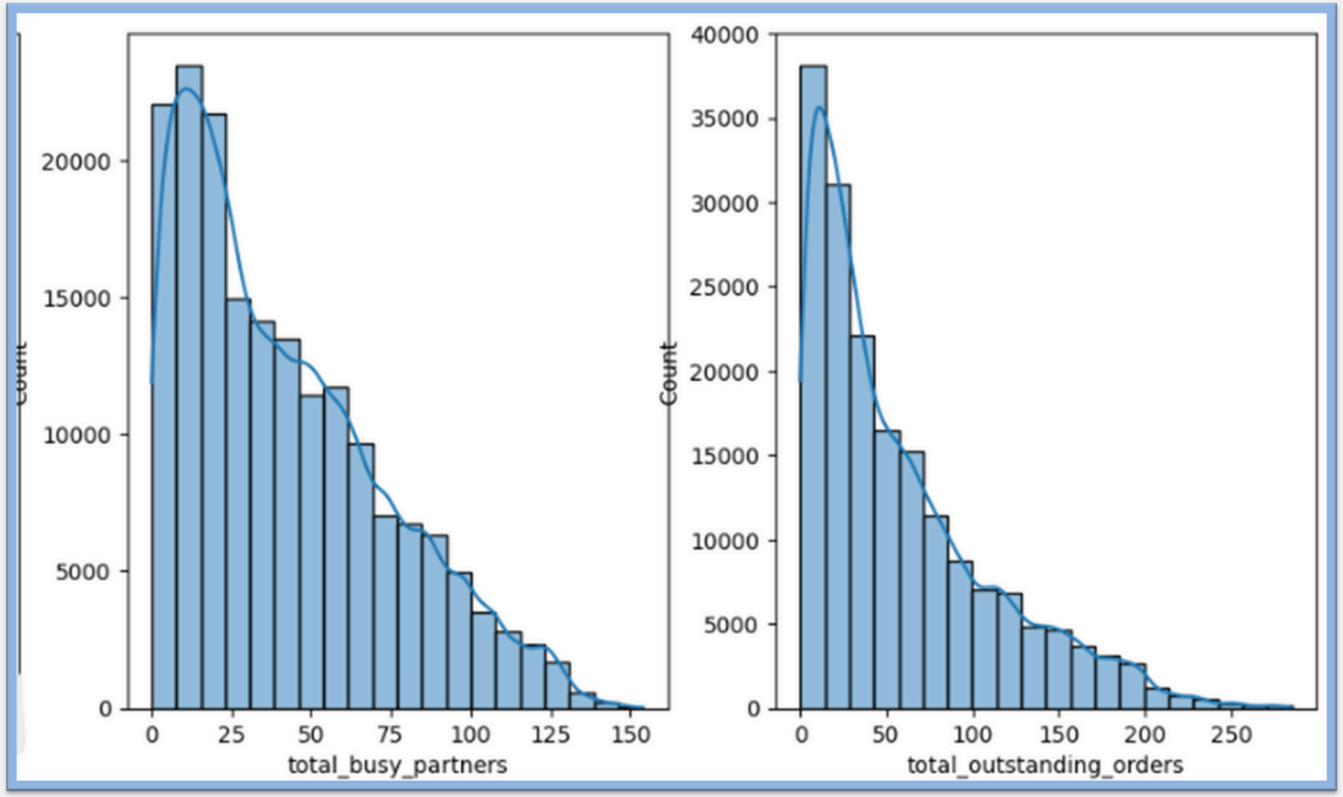


max_items_price:
The highest price per item predominantly falls within the range of Rs. 500 to Rs. 1,000.

total_onshift_partners:
The number of available couriers during peak order periods is often limited, as most are actively fulfilling ongoing deliveries.

Exploratory Data Analysis

01 Data Preparation & Preprocessing

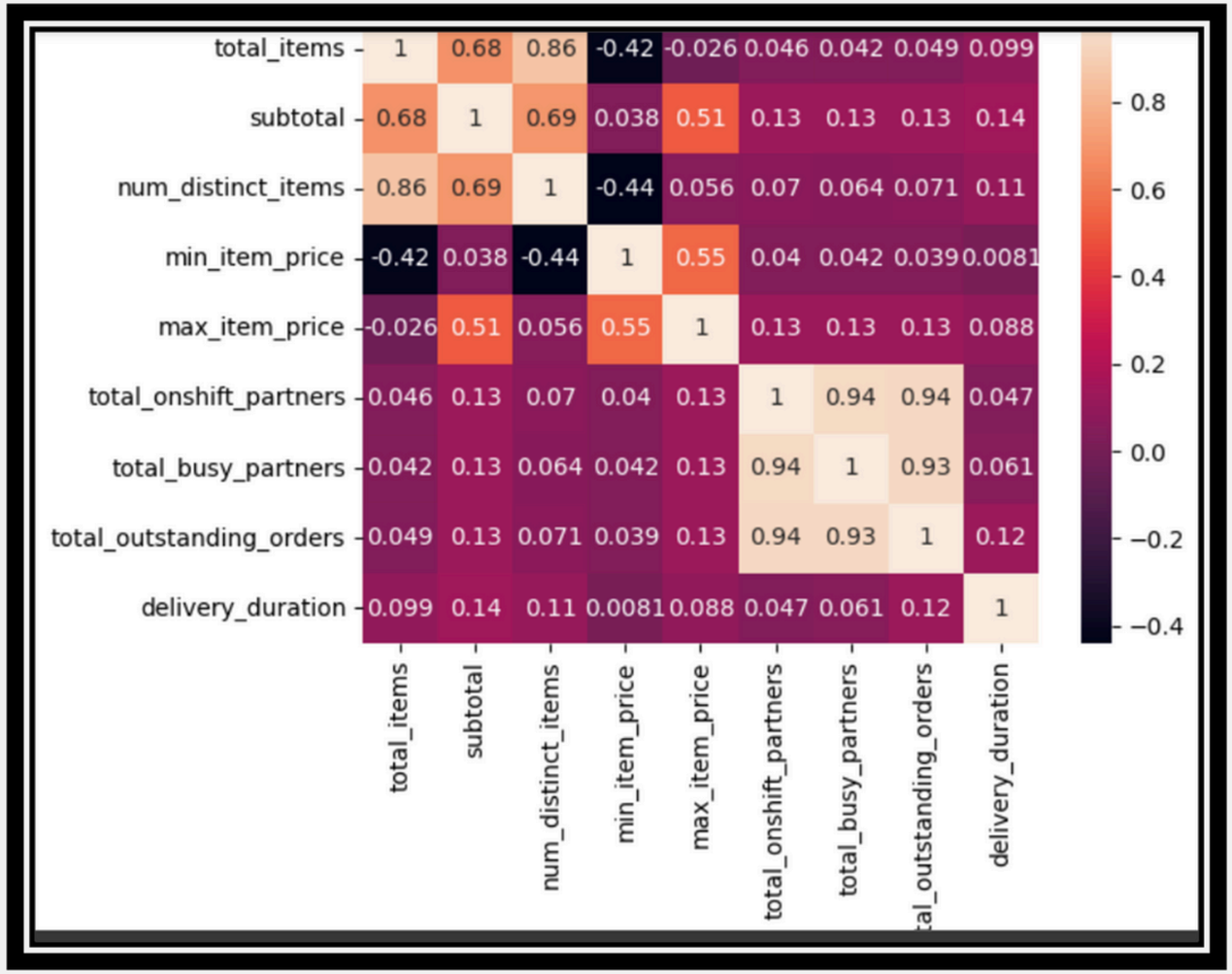


total_busy_partners:
When the number of orders is lower, more couriers are engaged in delivery activities.

total_outstanding_orders:
As the number of incoming orders increases, the number of orders that remain unassigned decreases.

Exploratory Data Analysis

01 Data Preparation & Preprocessing



total_items ↔ subtotal
The number of items ordered shows a 68% correlation with the total amount to be paid.

subtotal ↔ num_distinct_item
The total payable amount has a 69% correlation with the number of distinct item types ordered by the customer.

Data Cleaning, Imputation, and Encoding

01 Data Preparation & Preprocessing

Revalue was performed on categorical data (store_primary_category, order_protocol, market_id)

Filling null values with the mode (for categorical data)
Removing selected null values

Data encoding was applied to columns with categorical data types

Dropping columns that are not required for analysis and model training

01 Data Preparation & Preprocessing

Data Cleaning, Imputation, and Encoding

```
[ ]  # Copy data
     data_prepro = dataset.copy()

[ ]  # Mendeteksi kembali Missing Values
     data_prepro.isna().sum()

     market_id                    911
     created_at                     0
     actual_delivery_time           7
     store_primary_category      4182
     order_protocol               911
     total_items                    0
     subtotal                       0
     num_distinct_items             0
     min_item_price                 0
     max_item_price                 0
     total_onshift_partners         0
     total_busy_partners            0
     total_outstanding_orders       0
     delivery_duration              7
     dtype: int64
```

Rechecking for remaining null values

01

Data Preparation & Preprocessing

Data Cleaning, Imputation, and Encoding

```
# Mengelompokan data dalam kolom store_primary_category menjadi berbagai tipe restoran

def restaurant_category (store_primary_category):
  ethnic_based_restaurant = ['american','mexican','indian','italian','thai',
                             'chinese','singaporean','japanese','greek','filipino',
                             'asian','vietnamese','middle-eastern','persian',
                             'korean','latin-american','burmese','hawaiian',
                             'british','nepalese','peruvian','turkish','ethiopian',
                             'german','french','caribbean','afghan','pakistani',
                             'moroccan','malaysian','brazilian','european','african',
                             'argentine','irish','spanish','russian','southern',
                             'lebanese','belgian','mediterranean','cajun']

  specialize_food_restaurant = ['sandwich','salad','pizza','burger','barbecue',
                                'dessert','smoothie','seafood','steak','sushi',
                                'chocolate','pasta','alcohol','dim-sum','bubble-tea',
                                'tapas','soup','cheese']

  dietary_based_restaurant = ['vegan','vegetarian','gluten-free','kosher']

  other = ['cafe','catering','convenience-store','other','fast','breakfast',
           'comfort-food','gastropub','alcohol-plus-food']

  if store_primary_category in ethnic_based_restaurant:
    return 'Ethnic Based Food'
  elif store_primary_category in specialize_food_restaurant:
    return 'Specialize Food'
  elif store_primary_category in dietary_based_restaurant:
    return 'Dietary Based Food'
  elif store_primary_category in other:
    return 'Others'

# Mengaplikasikan kategori restoran tersebut kedalam fitur 'store_primary_category'
data_prepro.loc[: ,'restaurant_category'] = data_prepro['store_primary_category'].apply(restaurant_category)
```

Grouping data in the store_primary_category column into various restaurant types

01

Data Preparation & Preprocessing

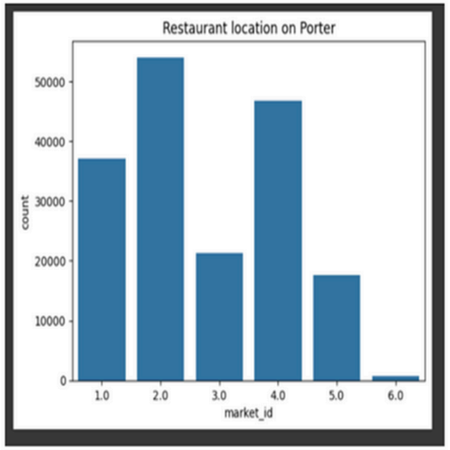
Data Cleaning, Imputation, and Encoding

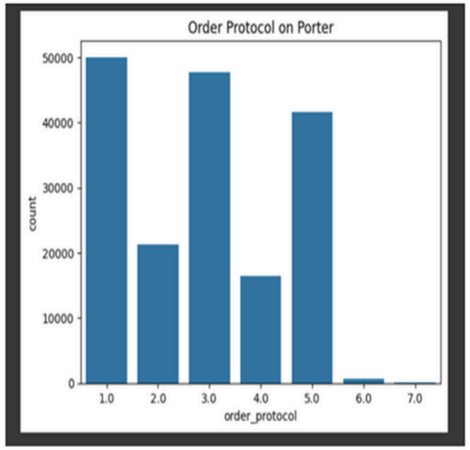
```
[ ]  # Memberikan penamaan ulang terhadap values dalam fitur order_protocol & market_id

     # Dari sumber data, penamaan valus dalam fitur 'order_protocol' adalah sebagai berikut:
     data_prepro.loc[:, 'order_protocol'] = data_prepro.loc[:, 'order_protocol'].replace({
         1.0 : 'Porter',
         2.0 : 'Call to Restaurant',
         3.0 : 'Pre-booked',
         4.0 : 'Third Party',
         5.0 : 'Others',
         6.0 : 'Others',
         7.0 : 'Others'
     })

     # Dari sumber data, penamaan valus dalam fitur 'market_id' adalah sebagai berikut:
     data_prepro.loc[:, 'market_id'] = data_prepro.loc[:, 'market_id'].replace({
         1.0 : 'Region 1',
         2.0 : 'Region 2',
         3.0 : 'Region 3',
         4.0 : 'Region 4',
         5.0 : 'Region 5',
         6.0 : 'Region 6'
     })
```

Renaming values within the order_protocol and market_id features

01 Data Preparation & Preprocessing

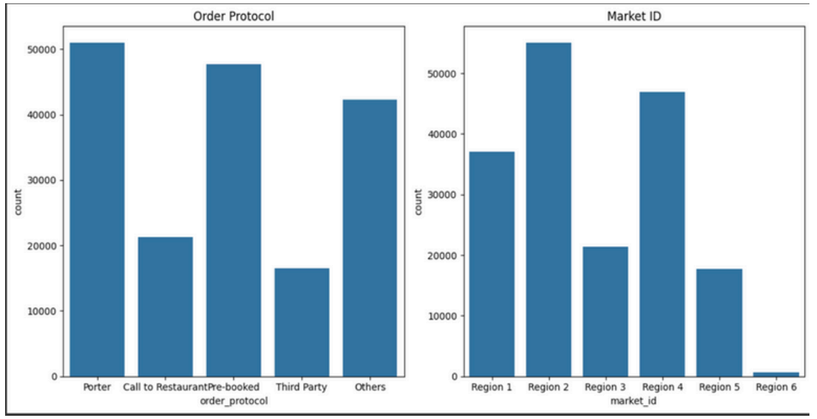
Data Cleaning, Imputation, and Encoding





Before

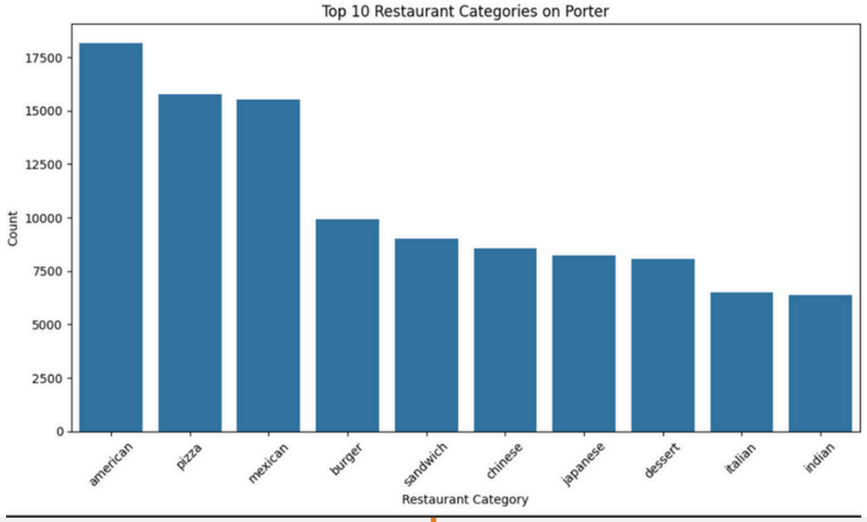
Differences before and after revaluation



After

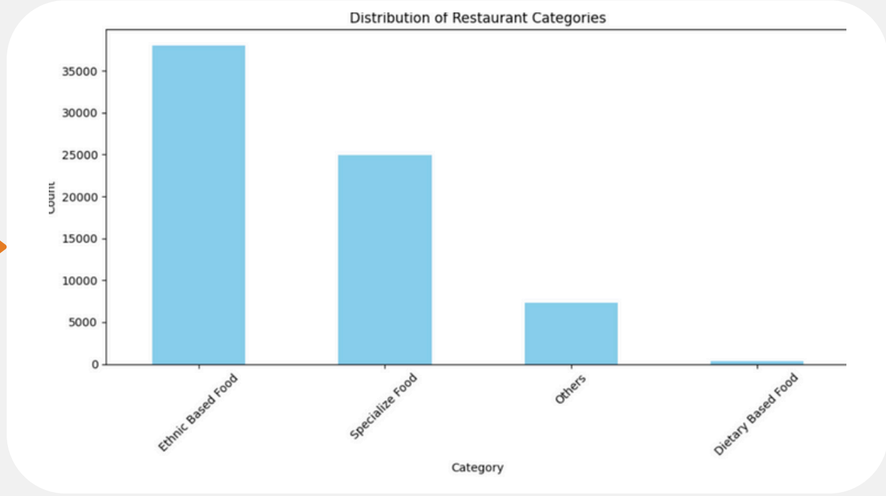
01 Data Preparation & Preprocessing

Data Cleaning, Imputation, and Encoding



Before

Differences before and after revaluation



After

01

Data Preparation & Preprocessing

Data Cleaning, Imputation, and Encoding

```
# Import ML Data Pre-Processing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, MinMaxScaler

# Melakukan Encoding untuk : market_id, store_primary_category, order_protocol
data_prepro_new = data_prepro.copy()

columns = data_prepro_new.select_dtypes(include=['object']).columns.to_list()

label_encoding = LabelEncoder()

# encode the data into a label
for i in columns:
  data_prepro_new[i] = label_encoding.fit_transform(data_prepro_new[i])
```

Encoding process for the following columns:

- market_id
- store_primary_category
- order_protocol

Label encoding is applied to convert categorical data types into numerical columns, enabling compatibility with machine learning models.

02 Data Modelling & Machine Learning

Data Scaling

In this stage, Min–Max Scaling is first applied to the features (X_train and X_test) to ensure that each feature's value range lies between 0 and 1.

Subsequently, regression models are trained using the scaled training data (X_train_scaled and y_train).

Data Modelling & Prediction

02 Data Modelling & Machine Learning

Regression models used and their definitions:

Linear Regression:
Linear regression is used to predict the dependent variable (y) based on the given independent variables (x).

Support Vector Regressor (SVR):
A supervised learning algorithm used to predict continuous variable values. It aims to find the optimal decision boundary (maximum-margin hyperplane).

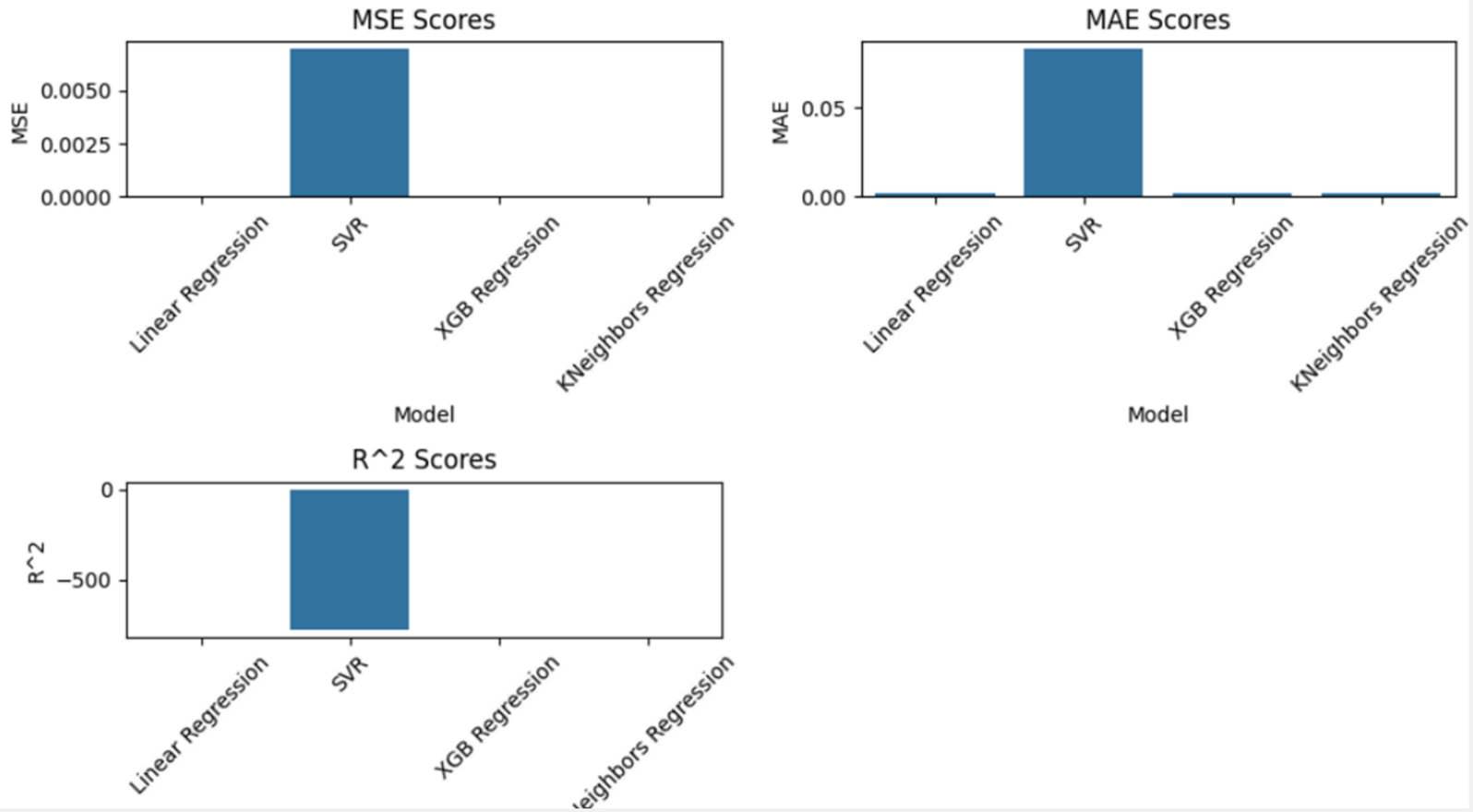
XGBRegressor:
A machine learning algorithm designed to make predictions on continuous numerical data.

K-Nearest Neighbor (KNN) Regressor:
A method that predicts values based on the closest data points in the training dataset, determined by distance to the target object.

02

Data Modelling & Machine Learning

Model Evaluation



02 Data Modelling & Machine Learning

Model Evaluation

Machine Learning	Model Evaluation		
	MSE	MAE	R2 Score
Linear Regression	0.0000075254002807	0.00196625596318019	0.159935790257553
SVR	0.00695302818382105	0.0832623345424095	-775.170025338984
XGB Regression	0.000009234656659	0.0018679308905265	-0.030869357092692
KNeighbors Regression	0.000011174144691669	0.0020438854500477	-0.247375379357164

Focusing on overall performance—particularly Mean Squared Error (MSE), which is the most commonly used metric for evaluating regression models—the Linear Regression model achieves the lowest MSE among all evaluated models.

Therefore, Linear Regression is selected as the model for subsequent use.

Model Evaluation

02 Data Modelling & Machine Learning

Mean Squared Error (MSE) and Mean Absolute Error (MAE):
These metrics measure how close the predicted values are to the actual values. **Lower values indicate better model performance.**

R^2 Score:
This metric indicates how well the model explains the variability in the data. An **R^2 value close to 1 signifies a strong model**, while a negative value indicates poor performance.

Based on these metrics, the evaluation is as follows:

- Linear Regression shows low MSE and MAE values with a positive R^2 score, indicating relatively strong performance.
- SVR exhibits high MSE and MAE values along with a highly negative R^2 score, reflecting poor performance.
- XGBoost Regression achieves low MSE and MAE values with a positive R^2 score, although it is lower than that of Linear Regression.
- KNN Regression records higher MSE and MAE values compared to Linear Regression and XGBoost Regression, along with a negative R^2 score.

Conclusion:
Based on the evaluation results above, Linear Regression and XGBoost Regression emerge as the strongest candidates, as both demonstrate low MSE and MAE values along with positive R^2 scores. However, **Linear Regression achieves a higher R^2 score compared to XGBoost Regression**. Therefore, Linear Regression appears to be the most suitable model for delivery time prediction given the available evaluation results.

Model Tuning

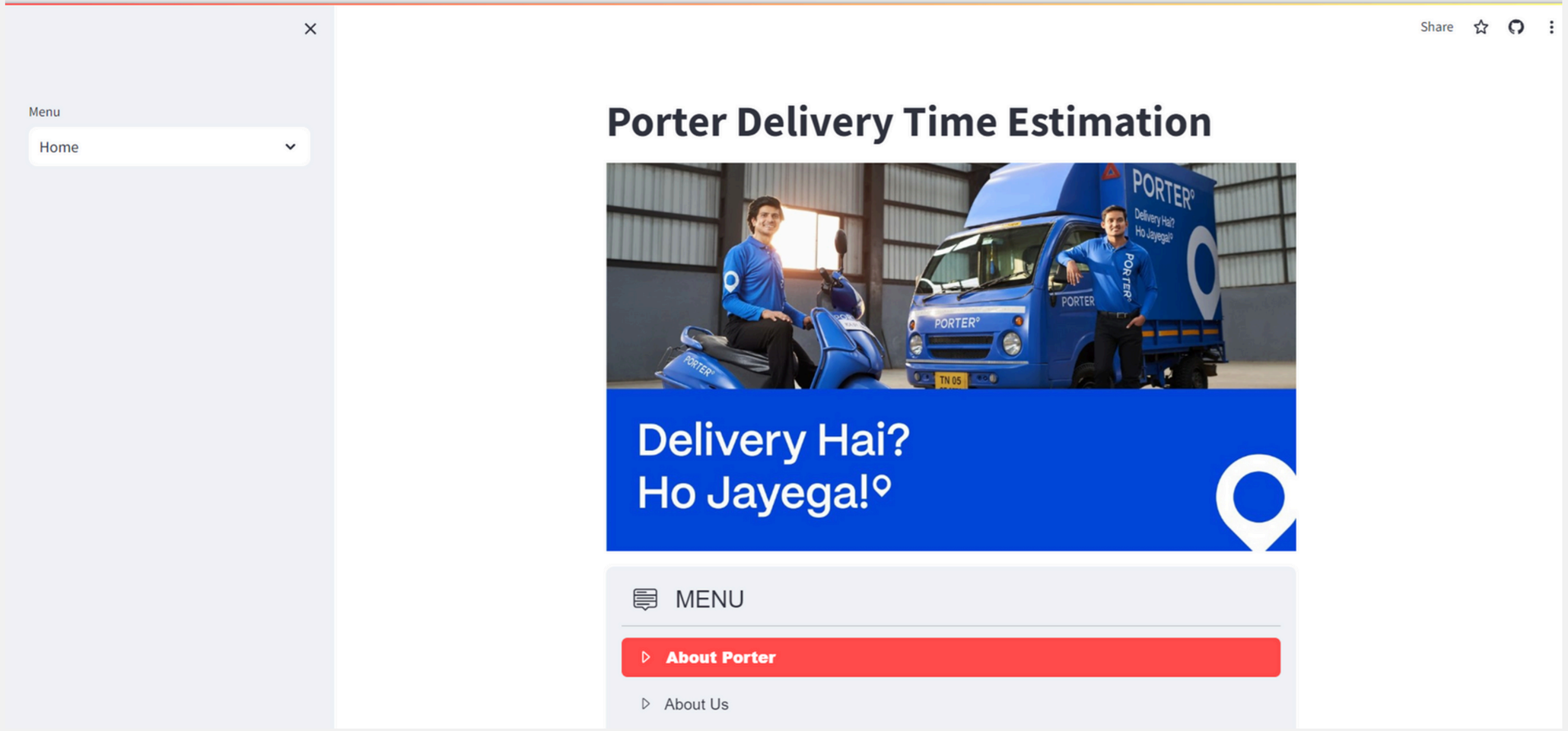
Machine Learning	Model Tuning
	Best Parameter Results
Linear Regression	Linear Regression {}
SVR	{'C': 10, 'gamma': 1, 'kernel': 'rbf'}
XGB Regression	{'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 300}
KNeighbors Regression	{'algorithm': 'brute', 'n_neighbors': 7, 'weights': 'uniform'}

Model tuning is an experimental process aimed at identifying the optimal hyperparameter values to maximize model performance. The tuning results of each model help determine the most suitable model to be selected.

03

Model Deployment

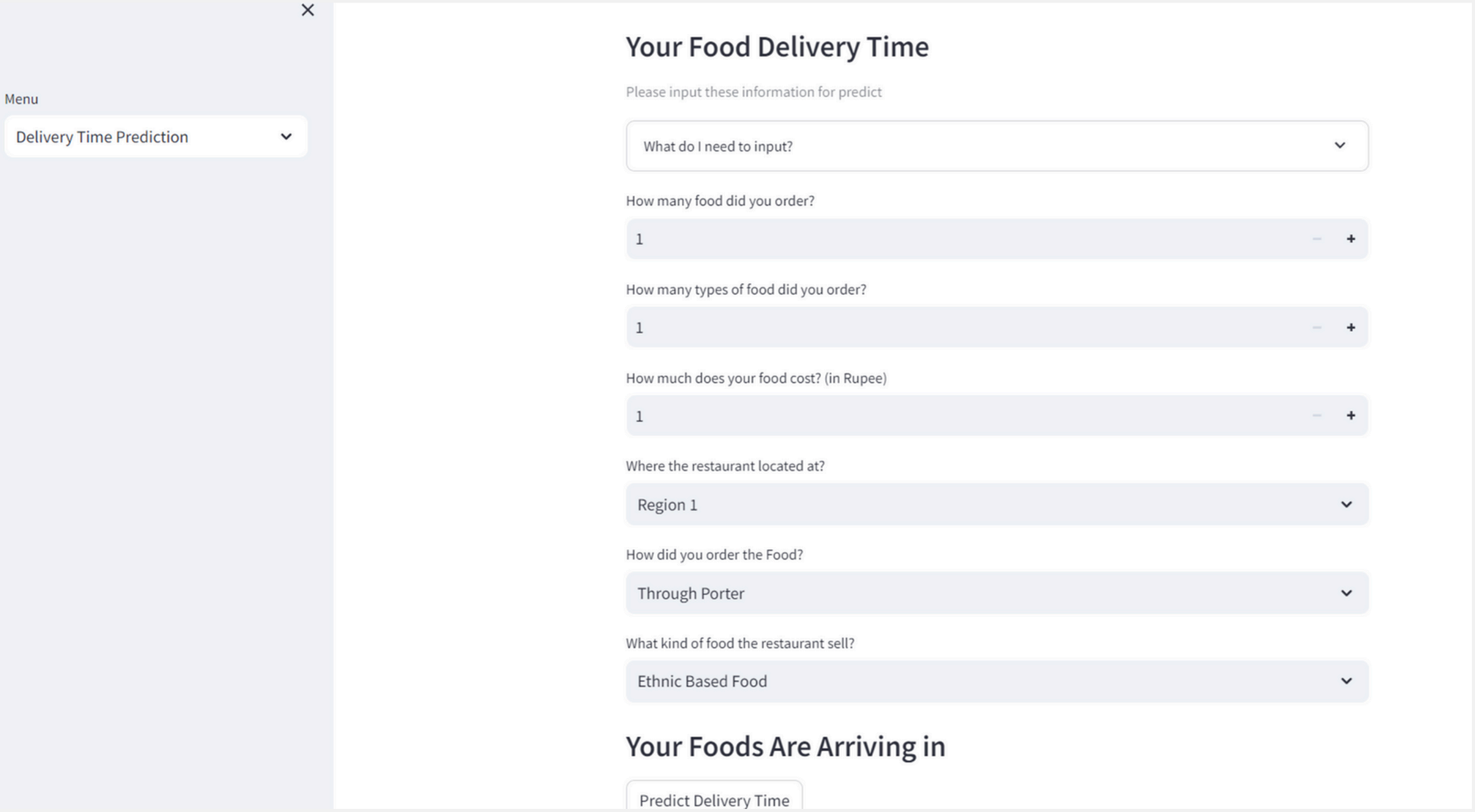
Deploying the model to Streamlit



03

Model Deployment

Deploying the model to Streamlit



03

Model Deployment

Deploying the model to Streamlit

How many food did you order?

4

How many types of food did you order?

4

How much does your food cost? (in Rupee)

4444

Where the restaurant located at?

Region 2

How did you order the Food?

Call to Restaurant

What kind of food the restaurant sell?

Others

Your Foods Are Arriving in

Predict Delivery Time

Estimated Delivery Time: [86.19688216] minutes

How many food did you order?

50

How many types of food did you order?

8

How much does your food cost? (in Rupee)

17865

Where the restaurant located at?

Region 6

How did you order the Food?

Pre-booked

What kind of food the restaurant sell?

Specialized Food

Your Foods Are Arriving in

Predict Delivery Time

Estimated Delivery Time: [317.31148688] minutes

Recommendations for **PORTER**

Dynamic Estimates
Leverage real-time data such as traffic and weather conditions.

Route Optimization
Utilize estimation models to determine optimal delivery routes.

Communication
Provide transparent and accurate delivery time estimates.

P**erformance Monitoring**
Monitor ML performance metrics, including averages and customer satisfaction.

Feedback Analysis
Analyze customer feedback and reviews to improve ML models.

Experimentation
Enhance ML performance by incorporating new data sources and exploring advanced algorithms.

Porter can leverage the delivery time estimation model to optimize operational efficiency and enhance the overall customer experience.

Links

GitHub
https://github.com/evanserlangga/Portfolio/tree/a4e795069ce3c0081bd57f395654c45a3cdb91f6/PORTER

Streamlit
https://final-project-honeycomb.streamlit.app/

Google Colaboratory
https://colab.research.google.com/drive/1N85nsxxAqvgG41sa94qwuhF9M29bwYN-?usp=sharing

Google Drive
https://drive.google.com/drive/folders/14iSx56rYd6-6-ElQ7Qf5BpRpAWXsTBS-?usp=sharing

Created By:
Ignatius Evans Erlangga

THANK YOU

My other Portfolios:

https://github.com/evanserlangga/Portfolio

https://drive.google.com/drive/folders/1x1dcO58XjpokD4KDw4I0xtwmtpckQgW3?usp=sharing